MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Memo No. 309 May 1974

COMMENTING PROOFS

James R. Geiser

ABSTRACT

This paper constitutes a summary of a seminar entitled "Commenting Proofs" given at the Artificial Intelligence Laboratory during the spring of 1974. The work is concerned with new syntactic structures in formal proofs which derive from their pragmatic and semantic aspects. It is a synthesis of elements from Yessenin-Volpin's foundational studies and developments in Artificial Intelligence concerned with commenting programs and the use of this idea in automatic debugging procedures.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract number N00014-70-A-0362-0003.

1. Introduction.

This paper constitutes a summary of a seminar entitled "Commenting Proofs" given at the A.I. lab at M.I.T. during the Spring of 1974. The work is concerned with new syntactic structures in formal proofs which derive from their pragmatic and semantic aspects. It is a synthesis of elements from Yessenin-Volpin's foundational studies (e.g. "Ultraintuitionism and the Antitraditional Program for the Foundation of Mathematics", Proceedings of the Summer Conference on Intuitionism and Proof Theory at Buffalo, New York, 1968) and developments in Artificial Intelligence concerned with commenting programs and the use of this idea in automatic debugging procedures (e.g. Gerald Sussman"s Doctoral Thesis: "A Computational Model of Skill Aquisition", M.I.T.,1973).

For the most part we shall restrict ourselves to the context of Peano Arithmetic and the primitive recursive arithmetic of addition, multiplication, and exponentiation in particular. At the end a few remarks will be made on how these ideas are to be extended to a richer deductive environment.

In our work we shall introduce means whereby it becomes possible to distinguish between formal proofs (a sequence of sentences satisfying the usual syntactical criteria) and "real" proofs (a formal proof constructed by someone with the goal of proving the theorem). In the latter case each line may be commented by intrinsic (formal-syntactic) and extrinsic (goal related) remarks. These remarks not only explain the purpose of a particular line, but at the same time

establish connections with other lines (previous lines and ones yet to be constructed)--in fact, these remarks point to connections between the very signs that make up the lines of the proof. These connections (which shall be called identificational connections or ids) are the links of a very detailed syntactic structure which resides implicitly in real proofs, a structure of "causal" chains connecting the occurences of symbols.

With this sort of information it becomes possible to answer a question like: "What part of the term 11·111 is responsible for the third stroke in the right hand side of the equation 11·111 = 111111?". Such a part will be called an ingredient of a term, consisting of a certain form of list structure whose atoms are the occurences of symbols in the term. Much of the present work is concerned with characterizing the dependence of ingredients on the computational paths used to evaluate a term. In the larger program of which the present work is a part, proofs are seen as the codification of procedures for manipulating list structures in such a way as to induce mappings from ingredients to ingredients. This provides a framework for understanding various traditional phenomena such as consistency and independence. On a higher level it suggests new ways to formalize such notions as relevant entailment and methods of proof. One last point before getting into the details. An equivalence relation is defined on ingredients so that every ingredient is equivalent to an ingredient in "normal" form. This form appears to be connected to a notion of computationally efficient computation paths.

2. Recursive Arithmetic.

Recursive Arithmetic consists syntactically of terms and equations, and

deductively of computations (contructed by means of the recursion axioms and the substitution rule).

Terms are expressions formed from 1,+,·,exp,(,) by means of the following rules.

1) 1 is a term.

2) If t is a term then so is t1.

3) If t and s are terms then so are (t + s), t·s, and exp(t,s).

We shall use the symbols $t,r,s,\ldots,t_1,r_1,s_1,\ldots$ to denote terms. We shall make use of the usual conventions for ignoring parenthesis. Also note that our use of the word "term" doesn't allow for the occurences of free variables, i.e. they are always to be closed. [1]

Terms of the form 1, 11, ..., $1^{(n)}$ (n concatenated strokes) are called numerals.

An equation is an expression of the form t = s.

The Recursion Axiom Schemata:

Addition	A1	(t + 1) = t1.
	A2	(t + s1) = (t + s)1.
Multiplication	M1	t·1 = t.
	M2	t·(s1) = t·s + t.
Exponentiation	E1	exp(t,1) = t.
	E2	exp(t,s1) = exp(t,s)·t.

---

1) The notational methods that we shall develop for +,·, and exp, will serve to handle all other primitive recursive functions.

The Substitution Rule Schema:

Line LA t = s(r)

Line LB r = r'

Line LC t = s(r').

Here s(r) denotes a term in which the term r has one or more indicated occurences, and s(r') denotes the term resulting from the replacement in s(r) of r by r' at the indicated occurences of r in s(r). Note that this is more general than uniform substitution. We say that LC follows from LA and LB by substitution.

A proof or computation is formally defined as a sequence of equations each of which is an instance of a recursion axiom or else follows from two previous equations by substitution. We consider two simple yet illuminating examples.

Example 2.1.

L1 1 + 11 = (1 + 1)1

L2 1 + 1 = 11

L3 1 + 11 = 111 .

Example 2.2.

L1 1·11 = 1·1 + 1

L2 1·1 = 1

L3 1·11 = 1 + 1

L4 1 + 1 = 11

L5 1·11 = 11 .

3. Commenting Proofs.

We begin commenting these computations by means of intrinsic and extrinsic remarks (this terminology is after Sussman). Then we use these remarks to generate ids between the occurences of symbols in the computation. By tracing out chains of ids we can establish an accountability for every sign in the computation. This will at the same time make explicit the semantics of the computation, i.e. which occurences of symbols are synonymous and what are the computational roles of each sign.

The intrinsic (or formal) comments make note of:

1) if the line is an axiom, in which case a pointer is generated to the axiom schema in question;

2) if the line follows by substitution, in which case pointers are generated to the lines from which it follows and to the occurences of the term to be replaced.

The extrinsic (or goal related) comments consist of:

1) the top level goal statement (i.e. to find the value of term t);

2) the assertion that line L (s = s') is generated in order to simplify by substitution a term t(s) in a previous line L';

3) the assertion that line L'' is the result of a substitution rule from lines L, L' whose purpose it was to achieve simplification by means of this substitution;

4) the assertion that the line matches the top level goal.

Example 2.1 Commented.

The top level goal is to evaluate 1 + 11.

L1 1 + 11 = (1 + 1)1 (Axiom A2) (Purpose is to simplify 1 + 11 of t.l.g.)

L2 1 + 1 = 11 (Axiom A1) (Purpose is to simplify 1 + 1 of r.h.s. of L1 using substitution.)

L3 1 + 11 = 111 (Sub. L1,L2) (Purpose is to fulfill the goal of L2. L3 matches t.l.g.)

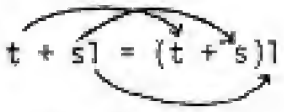
Example 2.2 is commented in a similar manner.

4. Identificational Connections.

We now add a third type of comment to the analyzed computation, namely we make note of the identificational connections (ids). First of all each axiom is to be accompanied by certain ids as follows.

A1 $t + 1 = t1$

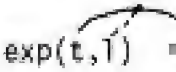
A2 $t + s1 = (t + s)1$



M1 $t \cdot 1 = t$

M2 $t \cdot (s1) = t \cdot s + t$

E1 $exp(t,1) = t$



E2 $exp(t,s1) = exp(t,s) \cdot t$.

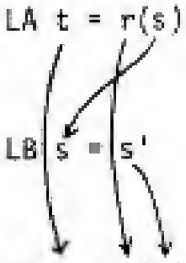
For example, in A2 we would say that t and s in the r.h.s. are re-written from the t and s in the l.h.s. and this justifies their synonymity. On the other hand we also make an identificational connection between the two occurences of 1 and this constitutes our semantical interpretation of + in terms of the sucessor function.

Now consider M2. We say that both t's in the r.h.s. are rewritten from

the l.h.s. t. Similarly the s in the r.h.s. is rewritten from the s in the l.h.s. However we associate the second t in the r.h.s. with the stroke 1 of the l.h.s.--this is part of our semantical interpretation of · . Thus we are looking at m·n as saying add m to itself n times; in this computation n acts as a counter for the n different rewritings of m. The strokes of n are control elements in this case.

Let A be an axiom and id(A) the set of ids associated with A. More explicily: if a stroke p on the r.h.s is simply rewritten from a stroke q on the l.h.s. (without a control element) then put id(p,q) in id(A); if, on the other hand p in the r.h.s. is rewritten from q in the l.h.s. under the control element q' and using an axiom for the operation f (either · or exp) then put id(p,(q,q',f)) in id(A).[1)]

Ids come from the substitution rule in accordance with the following schema.



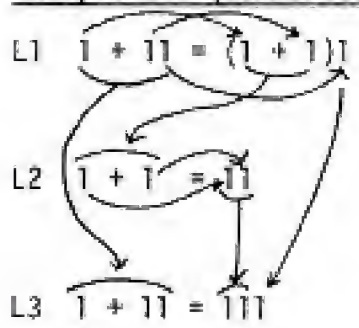
(These ids actually come from the extrinsic comments associated with these lines: LB's purpose is to simplify the indicated s in r.h.s. of LA. Hence the s in the l.h.s. of LB is rewritten from the s in the r.h.s. of LA. LC achieves the goal of LB; hence the context r in the r.h.s. of LC is rewritten from the context r in the r.h.s. of LA and the s' in the r.h.s. of LC is rewritten from the s' of the r.h.s. of LB. Also, the l.h.s. of LC is rewritten from the l.h.s of LA.)

---

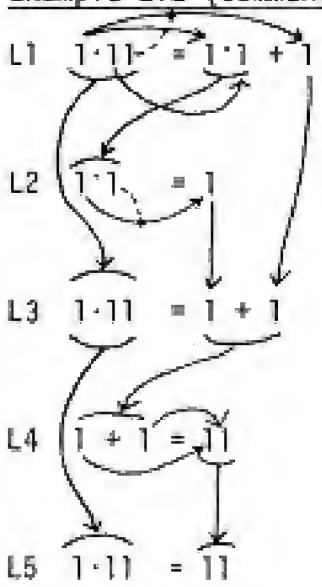
1) A set of ids like id(A) is usually taken symetrically,i.e., if id(u,v) is in id(A) then so is id(v,u).

Fully commented by ids Examples 2.1 and 2.2 look like the following.

Example 2.1 (Commented by ids.)



Example 2.2 (Commented by ids.)



We can now trace out paths of ids, thereby diagramming the computational relations between the different occurences of strokes in the proof. For example, in Example 2.1, if we denote the occurences of strokes in L3 by p,p',p",q, q',q" (left to right) then we can see that p is connected to q, p' is connected to q' and p" is connected to q". Furthermore these are the only connections between the strokes of L3. This yields a very nice correspondence between the l.h.s. and the r.h.s. strokes of L3.

The case of Example 2.2 is more complicated in that some of the occur-

rences of strokes act as counter elements and when in this role do not get rewritten. To present a complete analysis of ids in Example 2.2 we shall lable all occurences of strokes and to the right of each line we shall list the id associated with it. Between the lines we shall put the interline ids comming from the extrinsic comments.

L1 $1 \cdot 1\ 1 = 1 \cdot 1 + 1$ $\quad \{id(a_1,b_1),\ id(a_2,b_2),\ id((a_1,a_3,\cdot),b_3)\}$
$\quad a_1 \quad a_2a_3 \quad b_1 \quad b_2 \quad b_3$

$\{id(b_1,c_1),\ id(b_2,c_2)\}$

L2 $1 \cdot 1 = 1$ $\quad \{id((c_1,c_2,\cdot),d_1)\}$
$\quad c_1 \quad c_2 \quad d_1$

$\{id(a_i,e_i)$ for $i = 1,2,3,\ id(d_1,f_1),\ id(b_3,f_2)\}$

L3 $1 \cdot 1\ 1 = 1 + 1$ $\quad [id((e_1,e_2,\cdot),f_1),\ id((e_1,e_3,\cdot),f_2)]$
$\quad e_1 \quad e_2e_3 \quad f_1 \quad f_2$

$\{id(f_1,g_1),id(f_2,g_2)\}$

L4 $1 + 1 = 1\ 1$ $\quad \{id(g_1,h_1),\ id(g_2,h_2)\}$
$\quad g_1 \quad g_2 \quad h_1h_2$

$\{id(e_i,p_i)$ for $i = 1,2,3,\ id(h_1,q_1),\ id(h_2,q_2)\}$

L5 $1 \cdot 1\ 1 = 1\ 1$ $\quad [id((p_1,p_2,\cdot),q_1),\ id((p_1,p_3,\cdot),q_2)]$
$\quad p_1 \quad p_2p_3 \quad q_1q_2$

The ids in square brackets are derived from other ids. For example, $id((e_1,e_2,\cdot),f_1)$ of L3 is derived from:

$id((c_1,c_2,\cdot),d_1)$, $id(b_1,c_1)$, $id(b_2,c_2)$, $id(a_1,b_1)$, $id(a_2,b_2)$, $id(a_1,e_1)$, $id(a_2,e_2)$ and $id(d_1,f_1)$.

The rules to derive ids will be formulated in section 6.

Observe that it is possible to trace out a path from the stroke q_1 or q_2 of L5 to a pattern of strokes p_1,p_2,and p_3 of L5; this is exactly the contents of the square brackets accompanying L5. Thus we are lead to say that q_1 is rewritten from p_1 under the control of p_3. $(p_1,p_2,\cdot)$ and

$(p_1, p_3, \cdot)$ are the patterns in $|\cdot|$ of L5 "responsible" for q_1 and q_2 respectively in this computation. These patterns we call the ingredients of $|\cdot|$. Generally speaking when a term t is evaluated (i.e. proved equal to a numeral $|t|$) we may identify the ingredients of t responsible for each stroke in $|t|$. We want to show that the value of t is independent of the computation path. We shall also determine the way in which ingredients depend on the computation path.

5. Computation Paths.

On the previous pages we have presented two examples in some detail inorder to give an intuitive picture of what is happening with proofs, comments and ids. We shall now turn to a detailed study of the possible computation paths from a term.

We shall restrict our attention to computations which have the (standard) form:

L1 $t_1 = t_2$

.

.

.

L2i-1 $t_1 = t_{i+1}$

L2i $s_i = s_i'$

L2i+1 $t_1 = t_{i+2}$

.

.

.

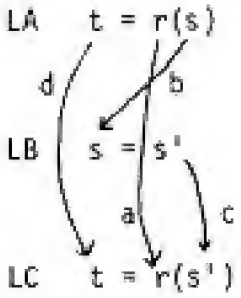
L2n-3 $t_1 = t_n$.

where L1 is an axiom and for $i = 1,\ldots,n-2$, L2i+1 follows from L2i-1 and L2i by substitution. We shall assume that these computations have been commented and the appropriate ids have been made.

The sequence of terms $t\ (=t_1)$, $t_2,\ldots t_n$ is called a computation path

from t to t_n.

Consider a step of the computation (called a simple reduction).



Define ID(r(s) → r(s')) to be the set of derived ids between the signs in r(s) and r(s'). Specifically:

1) If p is an occurence of a stroke in r(s') which is rewritten from q in r(s) via id a then id(p,q) is in ID(r(s) → r(s')).

2) If p is an occurence of a stroke in r(s') which is rewritten from q in s' of LB via c and q is rewritten from q' in s of the l.h.s. of LB,(i.e. id(q',q) is in id(s=s'),see p.7), and q' is rewritten from q" in s in the r.h.s. of LA via b , then id(p,q") is in ID(r(s) → r(s')).

3) If p is an occurence of a stroke in r(s') and is rewritten from q in s' of LB via c and q is rewritten from q' in s in the l.h.s. of LB under the control of q" in the l.h.s. of LB for the funtion f (i.e. id(q,(q',q",f)) is in id(s = s')), and q' and q" are rewritten from q° and $q^{\circ\circ}$ respectively in s in the r.h.s. of LA via b then id(p,(q°,$q^{\circ\circ}$,f)) is in ID(r(s)→ r(s')).

Every ocurrence of a stroke p in r(s') of LC is associated through ID(r(s)→ r(s')) with a unique stroke q in r(s) of LA or a unique pattern (q°,$q^{\circ\circ}$,f) of strokes q° and $q^{\circ\circ}$ in r(s).

If t (=t_1), t_2,...,t_n is a computation path P and t_n is a numeral then P is called an <u>evaluation path</u> and t_n is called the <u>value of t w.r.t.P</u>.

Define T(t) to be the set of all terms occuring in computation paths from t; the relation $s \rightarrow r$ determines a partial ordering of T(t) which we will now investigate.[1)]

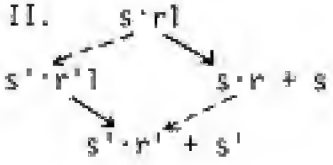
The notation $s \dashrightarrow r$ denotes a computation path from s to r. We say that $t_1 \dashrightarrow t_n$ is a subterm path if non of the reductions $t_i \rightarrow t_{i+1}$ involve the main function (outer most function symbol in polish notation).

Definition 5.1. A loop consists of two different computation paths going from a term r to a term s. A diamond is a loop which has either form I or II below. We illustrate this using · as the main function.

Form I.

(where the two indicated paths are subterm computation paths)

Form II.



(where both broken paths are subterm paths. We may assume that each involves the same associated subterm paths $s \dashrightarrow s', r \dashrightarrow r'$.

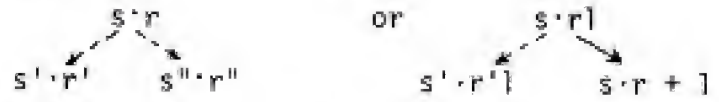
Definition 5.2. Two computation paths are simple variants if they differ by a diamond, i.e. they look like

Two paths P and Q are homologous if there is a sequence of computation paths P1,P2,...,Pn such that P = P1 and Q = Pn and for i = 1,...,n-1, Pi and Pi+1 are simple variants. Note that homologous evaluation paths assign the same value to a term.

---

1) If s = s' is an axiom then we shall also wright $s \rightarrow s'$, and call it a simple reduction. In this case ID($s \rightarrow s'$) is taken as id(s = s').

Lemma 5.3a. A split of the form



can be resolved into a diamond. (The dotted paths are subterm paths. Multiplication is just serving as a paradigm case.)

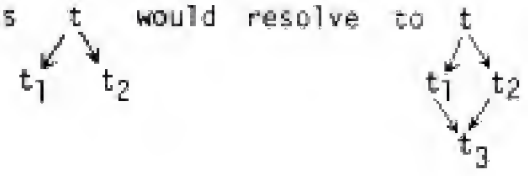
Lemma 5.3b. Any split can be resolved into a loop.

Lemma 5.3 is an anologue to the Church-Rosser theorem for the Lamda-calculus. The proof of this and other results is by induction on the rank $\underline{rk}(t)$ of a term, where $\underline{rk}$ is an integer valued function (primative recursive) defined so that a) if s is a subterm of t then $\underline{rk}(s) \lhd \underline{rk}(t)$, and b) if $s \rightarrow t$ then $\underline{rk}(t) \lhd \underline{rk}(s)$. The existence of such a function shows us that any computation path from t has less than or equal to $\underline{rk}(t)$ steps. If we were using all primitive recursive functions instead of just +, ·, and exp then such a rank function could be general recursive but not primitive recursive (e.g. like Ackerman's function). The organization of the proof is to dovetail 5.3a and 5.3b, first proving 5.3a for $\underline{rk}(t) = n$ and then 5.3b for $\underline{rk}(t) = n$.

Theorem 5.4. Any two evaluation paths for a term t are homologous.

Thus the value of a term t is independent of the evaluation path; $|t|$ will denote the value of t .

If we restricted substitution to uniform substitution then T(t) would have a much simpler form, namely splits t would resolve to



This would simplify much of our work. However since our eventual goal is the study of general proofs such a restriction would have to be relaxed.

6. Ingredients.

Definition 6.1. Let $q_1,\ldots,q_n$ denote the occurrences of strokes in the term t. Ing(t) denotes the set of list expressions obtained from $q_1,\ldots,q_n$, and $\cdot$,exp according to the following rules.

1) $q_1,\ldots,q_n$ are in Ing(t).

2) If $\underline{i}$ and $\underline{j}$ are in Ing(t) and f is $\cdot$ or exp then $(\underline{i},\underline{j},f)$ is in Ing(t).

The members of Ing(t) are called the abstract ingredients of t, and $q_1,\ldots,q_n$ are called the simple ingredients of t; $\mathrm{Ing}_0(t)$ is the set of simple ingredients of t. In $(\underline{i},\underline{j},f)$ $\underline{j}$ is called the control element and $\underline{i}$ is called the raw material.

Definition 6.2. $H : \mathrm{Ing}(t) \longrightarrow \mathrm{Ing}(s)$ is a homorphism (hom) iff for all $(\underline{i},\underline{j},f)$ in Ing(t),

$$H((\underline{i},\underline{j},f)) = (H(\underline{i}),H(\underline{j}),f).$$

Facts about homomorphisms:

6.3a. If $H,G : \mathrm{Ing}(t) \longrightarrow \mathrm{Ing}(s)$ are homomorphisms which agree on $\mathrm{Ing}_0(t)$ then H = G.

6.3b Any map $H : \mathrm{Ing}_0(t) \longrightarrow \mathrm{Ing}(s)$ extends to a unique hom from Ing(t) into Ing(s).

6.3c Hom H is 1-1 on Ing(t) iff H is 1-1 on $\mathrm{Ing}_0(t)$.

We associate with a simple reduction $t \to s$ the following hom $H[t \to s]$. Let $ID(t \to s)$ be the set of ids accompanying $t \to s$ as defined on p.11. $H[t \to s]$ is the unique hom from $Ing(s)$ into $Ing(t)$ determined by (for q in $Ing_0(s)$)

$$H[t \to s](q) = \begin{cases} p & \text{if } id(p,q) \text{ is in } ID(t \to s), \\ (p,p',f) & \text{if } id((p,p',f),q) \text{ is in } ID(t \to s). \end{cases}$$

$H[t \to s]$ is 1-1 on $Ing_0(s)$ and hence is 1-1 on all of $Ing(s)$.

Definition 6.4. Let $P = t_1,\ldots,t_n$ be an evaluation path for t; thus $|t| = t_n$. Define the hom $H[P] = H[t_1 \to t_2] \circ H[t_2 \to t_3] \circ \cdots \circ H[t_{n-1} \to t_n]$. The set of

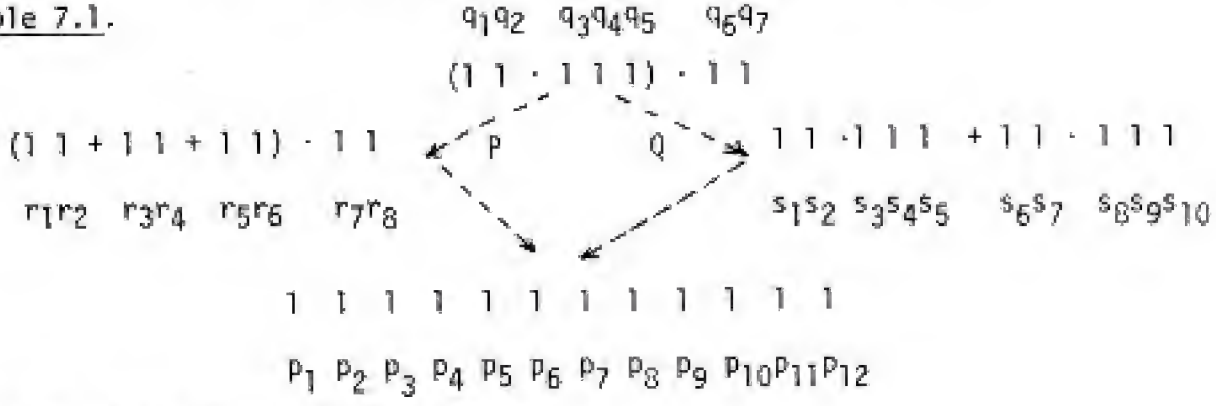
(real)ingredients w.r.t. P is the set $H[P](Ing_0(t))$ and is denoted by $Ing(t;P)$.

Note that $H[P]$ is a 1-1 map from $Ing_0(|t|)$ into $Ing(t)$. So the cardinality of $Ing(t;P)$ is equal to the integer denoted by the numeral $|t|$. We think of $H[P](q)$ as the unique "computational pattern" in t which is "responsible" for q via P.

7. "Invariance" of Ingredients.

How does $H[P]$ depend on P? Consider the following example.

Example 7.1.



Computing $H[P](p_9)$ and $H[Q](p_9)$ we get

$H[P](p_9) = \underline{i} = ((q_1,q_4,\cdot),q_7,\cdot)$ and $H[Q](p_9) = \underline{j} = ((q_1,q_7,\cdot),(q_4,q_7,\cdot),\cdot)$.

The basic difference between $\underline{i}$ and $\underline{j}$ is the temporal order in which the control elements q_4 and q_7 are operating and not in the ultimate control relationships.

The relation between $\underline{i}$ and $\underline{j}$ is made into a basic equivalence relation.

Definition 7.2. Let $\underline{i}$ and $\underline{j}$ be ingredients in Ing(t). $\underline{i} \equiv \underline{j}$ iff there is a sequence of pairs $(\underline{i}_1,\underline{j}_1),\ldots,(\underline{i}_n,\underline{j}_n)$ of ingredients in Ing(t) (this sequence would be called a derivation of $\underline{i} \equiv \underline{j}$) such that $\underline{i} = \underline{i}_n$, $\underline{j} = \underline{j}_n$, and for $k = 1,\ldots,n$, either

1) $\underline{i}_k = \underline{j}_k$,

2) there exists ingredients $\underline{u},\underline{v},\underline{w}$ and function symbols f,g such that
$\{\underline{i}_k, \underline{j}_k\} = \{((\underline{u},\underline{v},f),\underline{w},g), ((\underline{u},\underline{w},g),(\underline{v},\underline{w},g),f)\}$,

3) there exists k', k" less than k such that
$\underline{i}_k = (\underline{i}_{k'},\underline{i}_{k"},f)$ and $\underline{j}_k = (\underline{j}_{k'},\underline{j}_{k"},f)$.

7.3. Facts about $\equiv$.

7.3a) $\equiv$ is an equivalence relation on Ing(t).

7.3b) If $\underline{i} \equiv \underline{j}$ and $\underline{i}' \equiv \underline{j}'$ then $(\underline{i},\underline{j},f) \equiv (\underline{i}',\underline{j}',f)$.

7.3c) If $\underline{i}_k \equiv \underline{j}_k$ for k = 1, 2, 3 then
$$((\underline{i}_1,\underline{i}_2,f),\underline{i}_3,g) \equiv ((\underline{j}_1,\underline{j}_3,g),(\underline{j}_2,\underline{j}_3,g),f).$$

Theorem 7.4. Let $H : Ing(s) \rightarrow Ing(t)$ be a homomorphism. Then H preserves $\equiv$, i.e., if $\underline{i}$ and $\underline{j}$ are in Ing(s) and $\underline{i} \equiv \underline{j}$ then $H(\underline{i}) \equiv H(\underline{j})$.

Theorem 7.5. Let $G,H : Ing(s) \rightarrow Ing(t)$ be two homomorphisms and suppose that for all p in $Ing_0(s)$, $G(p) \equiv H(p)$. Then $G(\underline{i}) \equiv H(\underline{i})$ for all $\underline{i}$ in Ing(s).

A subterm computation P: $s\cdot t \rightarrow s'\cdot t'$ can be decomposed in a natural

way into two computation paths P(s) : s-->s' and P(t) : t-->t'. If we identify the strokes of s' with the corresponding strokes of s' in s'·t' then we may consider Ing(s') as a subset of Ing(s'·t'),and likewise for t'. The hom H[P] may then be decomposed into two homs H[P(s)] and H[P(t)]. We formulate this in the next lemma.

Lemma 7.6. Let P, P(s) and P(t) be as above. then

H[P] restricted to Ing(s') equals H[P(s)], and

H[P] restricted to Ing(t') equals H[P(t)].

An ingredient in Ing(t) is called a real ingredient if it is equal to H[P](q) where P is some evaluation path from t and q is in $Ing_0(|t|)$.

Theorem 7.7. (Invariance of Ingredients.) Let P and Q be two evaluation paths for t. Then for all p in $Ing_0(t)$, H[P](p) ≡ H[Q](p).

This is an immediate consequence of the next lemma.

Lemma 7.8.Let t be any term with rk(t) less than or equal to n.

I. If Pand Q are computation paths from t to t' forming a diamond and $\underline{i}$ in Ing(t') is real then H[P]($\underline{i}$) ≡ H[Q]($\underline{i}$).

II. If P and Q are computation paths from t to t' and $\underline{i}$ is in Ing(t') and is real then H[P]($\underline{i}$) ≡ H[Q]($\underline{i}$).

The lemma is proved by induction on n and dovtailing I and II. Lemma 5.3 plays a major role.

8. Final Remarks.

The next step in this investigation extends our analysis to general

proofs in first order logic and full Peano Arithmetic. This means associating with logical axioms the appropriate sets of ids (e.g. consider $A \supset (B \supset A)$) and determining the interline ids associated with the extrinsic commenting of proofs. These ids will provide the means of constructing list manipulating procedures from proofs which will give rise to semantic interpretations for these proofs.

It is also striaghtfoward to apply these ideas to other deductive-computational systems such as the lamda-calculus and Curry's Combinatorial Logic.